Rapid Releases and Testing Problems at the industry: A survey

Maximiliano A. Mascheroni[1,2], Emanuel Irrazábal[2], Juan A. Carruthers[2], Juan A. Pinto[2]

[1] Facultad de Informática. Universidad Nacional de La Plata. La Plata, Buenos Aires, Argentina.
[2] Facultad de Ciencias Exactas y Naturales y Agrimensura. Universidad Nacional del Nordeste. Corrientes, Argentina.

{agustin.mascheroni, emanuelirrazabal}@gmail.com
{juan.carruthers, juan_al_pinto}@hotmail.com

Abstract. Rapid releases and continuous software development are established practices in modern agile projects. The advantages of them are widely known across the software development community, but there are some studies which mention that there are still challenges to face. According to them, there are different open issues which are affecting the implementation of an adequate testing process. With the aim of validating if these problems are present in real projects, in this paper we present the results of a survey whose goal was to validate whether the industry is experiencing similar issues and their causes. The findings demonstrate that both the industry and academic side are aligned, and that there is still a need for processes and tools regarding the testing process in continuous development.

Keywords: Continuous testing, continuous software development, continuous delivery, continuous deployment, survey.

1 Introduction

Currently, there are many companies which develop software that are moving from traditional release cycles to rapid releases. Practitioners of continuous deployment (CDP) or continuous delivery (CD) concepts, claim that deploying software to production continuously offers various benefits to companies and their end users [1–4]. However, according to some authors, there are still some challenges to face [5–11, 12, 13]. One of these claims is the relation between rapid release models and software quality [14]. Examples of software quality issues are: not enough time for testing [15], slow bug fixing [16], small scope for the testing stage [17].

With the goal of identifying any type of problems related to software quality in continuous development environments, several studies have been carried out by many authors. It has been found that the most reported issues are: *time-consuming testing; flaky tests; ambiguous test results; GUI automated testing; dynamic Web UI*

automated testing; data testing; big data testing; mobile automated testing, non-functional automated testing; automated testing of applications composed by cloud services; testing as a service; and *webservices automated testing* [11, 18, 19].

Also, in [18] it has been analyzed the relationship between these problems. It has been found that the root problems are time-consuming testing, flaky tests and lack of frameworks for testing certain type of applications. Fig 1 shows this relationship.

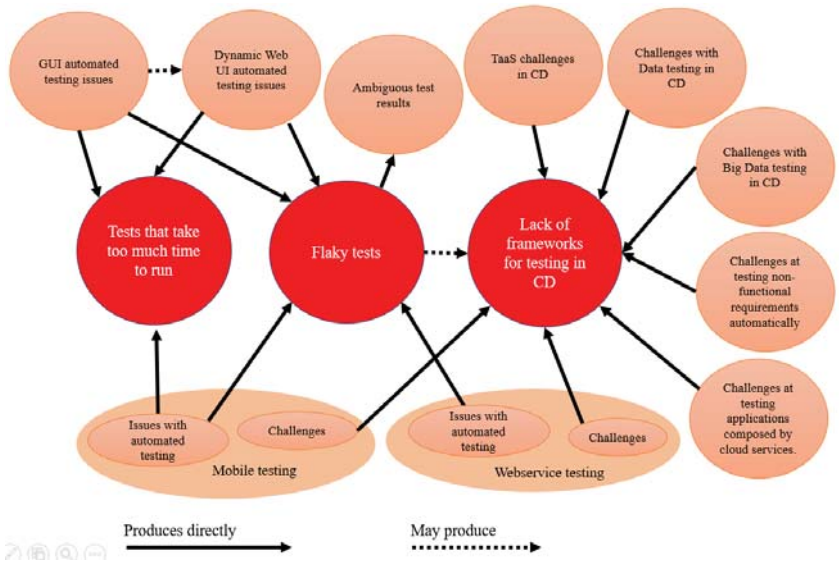

Fig. 1. Relationship between CD testing problems [18].

As it was aforementioned, the findings of the studies show that the more important testing-related issues in rapid releases are time-consuming testing, flaky tests and lack of frameworks. However, those results were taken only from academic sources which are not directly related to the industry. Thus, this research aims to validate whether the industry is facing the same problems, they have never experienced them, or they have already got rid of them.

Apart from this introductory section, in Section 2 it is described the survey and its components. The findings of the survey are presented in Section 3, which are discussed later in Section 4. Finally, conclusions and future work are highlighted in Section 5.

2 The survey

This research employed a survey as the main data gathering method, following Kitchenham's principles of survey research [20–25]. In this section, it is described the procedure followed to design and conduct the study.

2.1 Research goals

The first step was setting the goals which were derived from previous studies. In the context of rapid releases at the industry, the goals of this research are:

- Validate if the industry is having the same testing problems reported by the academic sources.
- Detect new challenges and open issues at implementing the testing process.

2.2 Survey design and development

The survey is a cross sectional survey [26], because it is necessary to get a snapshot about the testing process in the present. The survey instrument was the questionnaire, and after the design of it, a pilot process was performed in order to test the validity and readability of the questions. As part of it, project managers and team mates supplied feedback for improving the survey. Based on that feedback, and after making the modifications to the survey, the process was repeated with other project managers.

The target population was companies that have projects and teams which work with rapid releases. However, as it is impossible to send a questionnaire to all of the continuous software development projects around the world, based on similar surveys that have a sample size in a range between 100 and 250 respondents [27–32], the sample size for this research was 255 projects. The method for sharing the questionnaire was the use of social networks. It was provided a briefly introduction where respondents were able to see the goals of the study, the value of their participation and how they were going to be benefited from it. According to Kitchenham [23], respondents will be more motivated to supply complete and precise responses if they can see that the results are likely to be useful to them.

There were four sections in the survey, where the first section was the introductory part with the purpose of the study, concepts and clarifications. The second section was on metadata (project information) such as type of software being developed, the number of members in the team, their roles and timebox duration. The third section was on the use of continuous development practices, which results will be used for other studies. The fourth section was on problems, challenges and solutions in the testing process using open questions. The reason for open questions is because they allow respondents to describe all the problems they have and the solutions or workarounds they implemented. The survey can be found in the following link: https://forms.gle/8ULNMgVxzone3WvE6.

3 Results

In this section, the findings of the survey are presented. During a period of four months, a total of 287 responses were obtained. However, after rejecting results with incomplete questionnaires and with answers that were not consistent each other, a total of 255 projects was gotten.

The projects were grouped based on the size of the team and the type of application being developed (see Fig. 2). It is very important to highlight that most of the teams are working in more than one platform. For the team size, it was used a classification proposed by Yang et al. [33], based on O'Connor and Yang metrics [34], as follows:

- Small team: < 16 members.
- Medium team: 16 – 45 members.
- Large team: > 45 members.

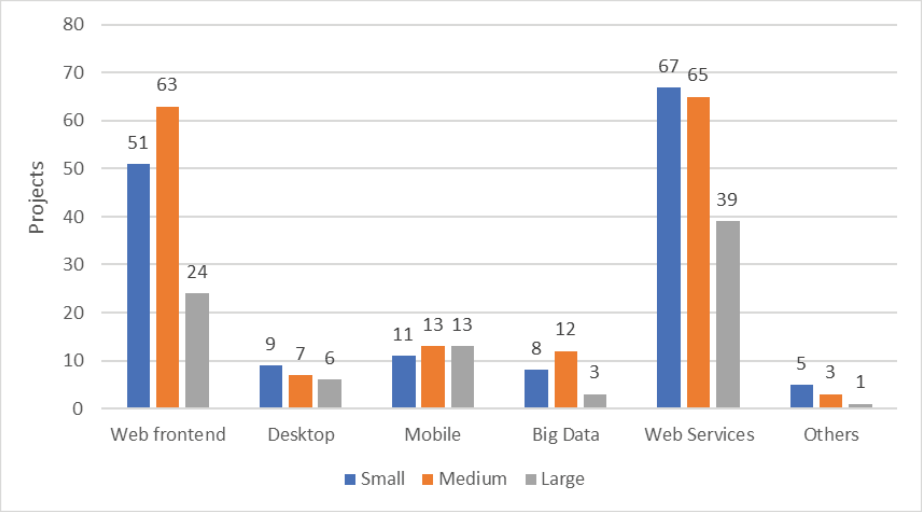

Fig. 2. Number of projects grouped by team size and platform being developed.

3.1 Testing Problems at the industry

As it was described in Section 1, the literature reports that there are some problems related to the testing in continuous development environments. The purpose of this survey was to validate whether the industry is facing the same issues or not. The results obtained from the survey can be seen in Table 1 and Fig. 3.

As it can be seen in Table 1, most of the outlined problems match with the ones reported by the literature. There were also new problems detected:

1. Lack of procedures, patterns and good practices for automated testing in continuous development.
2. Unstable environments.

Flaky test was the problem with greater occurrence (224 times). According to the companies, there are several reasons for having flaky tests such as missing or incorrect test data, GUI components that cannot be found, timeouts due to network issues or unstable environments, dynamic GUI elements which take long to load and inconsistence test code. Regarding GUI automated testing, the most reported issue

was maintaining the test code at changes on the UI element's locators. Furthermore, GUI automated tests are tests that take much longer time to run that any other test. This issue was reported by 179 companies. In the same way, running automated scripts on a page which has dynamic components is a challenge for 47 projects, because they have mentioned that tests fail when those elements are not shown yet. Time-consuming testing affects also mobile (17) and web service testing (58).

Table 1. Testing problems found by the survey.

Problem	N° of projects
Flaky tests	224
Time-consuming testing	179
GUI automated testing	124
Lack of procedures, patterns and good practices for automated testing in rapid releases environments	112
Ambiguous test results	65
Web service testing	58
Dynamic Web UI automated testing	47
Unstable environments	36
Data testing	26
Non-functional automated testing	23
Big Data testing	18
Mobile testing	17
Automated testing of applications composed by cloud services	2

A group of 26 people have reported that testing data attributes automatically is also a challenge. Similarly, there are some projects implementing big data tools and frameworks and most of them have stated that there is a lack of information, documentation or guides on the web about big data testing. These projects mentioned the need for verification mechanisms at ETL stages and data analysis. Other companies have outlined issues with automated testing in mobile applications, where they highlight the lack of robust frameworks, processes and models, especially for running the tests continuously. The same problems are faced by teams which work on web services or that develop applications with cloud services.

For testing non-functional requirements, 17 projects were not able to integrate performance test scripts on a CI pipeline. Another 6 teams reported issues with lack of documentation and tools about security testing on CI, CDP or CD. Also, according to 65 people, understanding the reports is an issue because lack of details about failures.

Finally, 112 projects stated in different ways that there is a need for procedures, patterns and good practices for automated testing in rapid releases environments. For example, some teams mentioned that there is lack of good practices for getting a good coverage of automated tests for different layers in the application. Other projects have stated that there is a lack of standards about test data management strategies and test automation framework architecting. Some respondents mentioned that while there are thousands of tutorials, trainings, blogs and courses about traditional test automation, there is lack of them about continuous test automation (continuous testing).

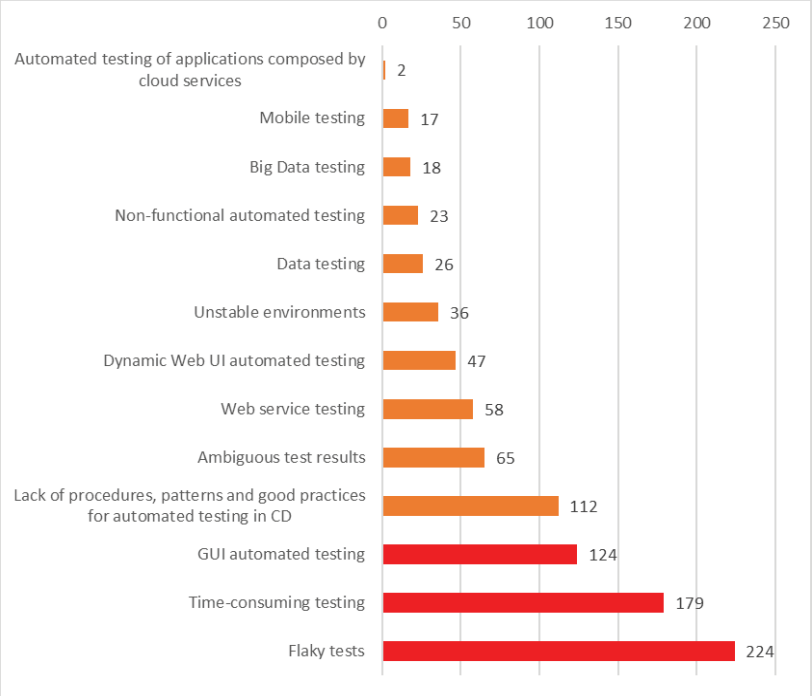

Fig. 3. Reported problems.

4 Discussion

In the last section the results found in the survey have been presented. From there, a deeper analysis can be performed, and different conclusions can be taken. In this section, the findings of the survey will be analyzed by discussing the reported testing problems.

The most reported problems were flaky automated tests, tests that take long to run, automated tests for GUI applications and lack of frameworks, tools, procedures and documentation for automated testing in rapid releases. The metadata obtained from the second part of the survey will be used to analyze each testing problem. Thus, the analysis is used later for determining whether a problem reported by the literature is also valid for the industry or not.

Table 2 presents the number of testing problems by the type of platform being developed using the following annotation:

- P1: Flaky tests
- P2: Time-consuming testing
- P3: GUI automated testing
- P4: Lack of procedures, patterns and good practices for automated testing in rapid releases environments

- P5: Ambiguous test results
- P6: Web service testing
- P7: Dynamic Web UI automated testing
- P8: Unstable environments
- P9: Data testing
- P10: Non-functional automated testing
- P11: Big Data testing
- P12: Mobile testing
- P13: Automated testing of applications composed by cloud services

Table 2. Number of testing problems by type of platform being developed.

Platform	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	P11	P12	P13
Web frontend	134	126	92	46	34	0	47	18	12	12	0	0	1
Desktop	17	19	17	16	7	0	0	4	2	2	0	0	0
Mobile	19	29	14	12	5	0	0	6	1	2	0	17	0
Big Data	1	1	0	15	0	0	0	1	1	1	18	0	0
Web Services	48	3	0	20	19	58	0	7	8	4	0	0	0
Other	5	1	1	3	0	0	0	0	2	2	0	0	1
Total	224	179	124	112	65	58	47	36	26	23	18	17	2

Flaky tests (P1), time-consuming testing (P2) and lack of frameworks (P4) were reported in all the different type of projects. Data testing (P9) and non-functional automated testing (P10) were also outlined by all of them, but they just represent a low percentage of the total: P9 is 10% (26 over 255) and P10 is 9% (23 over 255), where 255 is the total of projects. Using the same formula, P1 is 87%, P2 is 70% and P4 is 43%.

GUI automated testing (P3) was reported by all the teams except the ones that work on big data and web service-only applications. Thus, the total of projects that will be considered are 198 (as the sum of all the platforms except big data and web service projects). After applying the mentioned formula, P3 is 62% (124 over 198).

Using the same criteria: web service testing (P6) is 34% (58 over 171), dynamic Web UI automated testing (P7) is 34% (47 over 138), big data testing (P11) is 78% (18 over 23), mobile testing (P12) is 45% (17 over 37), and automated testing of applications composed by cloud services is 1% (2 over 147).

As regards of unstable environments (P8) and ambiguous test results (P5), not all the teams have reported this problem. However, on the one hand, a report is an artifact of any type of test execution (no matter the platform). On the other hand, any type of test runs on a test environment which may be unstable or not. Thus, those problems are not tied to a certain type of technology. Therefore, P5 is 25% (65 over 255) and P8 is 14% (36 over 255).

Finally, Table 3 shows the severity of each problem reported by the literature at the industry, using the following criteria: 0% is not an issue; 0,1% to 19,9% is a very low severity issue; 20% to 39,9% is a low severity issue; 40% to 59,9% is a medium severity issue; 60% to 79,9% is a high severity issue; and finally 80% to 100% is a critical issue.

Table 3. Relationship between the testing problems in the literature and the industry.

Problem	Reported by the literature	Severity at the industry
Flaky tests	Yes	Critical (87%)
Time-consuming testing	Yes	High (70%)
GUI automated testing	Yes	High (62%)
Lack of procedures, patterns and good practices for automated testing in rapid releases environments	No	Medium (43%)
Ambiguous test results	Yes	Low (25%)
Web service testing	Yes	Low (34%)
Dynamic Web UI automated testing	Yes	Low (34%)
Testing as a service (TaaS)	Yes	Not an issue (0%)
Unstable environments	No	Very Low (14%)
Data testing	Yes	Very Low (10%)
Non-functional automated testing	Yes	Very Low (9%)
Big Data testing	Yes	High (78%)
Mobile testing	Yes	Medium (45%)
Automated testing of applications composed by cloud services	Yes	Very Low (1%)

5 Conclusions and future works

In this paper it has been presented the findings of a survey whose main goal was to validate whether the industry is experiencing the same testing problems that academic sources have reported.

Most of the literature's testing problems match with the ones reported by the organizations which have participated. The only issue that was not reported by these companies was Testing as a Service (TaaS). On the other hand, two new issues have been outlined by the respondents: unstable environments, and lack of frameworks such as procedures, good practices, models, and patterns for continuous automated testing in rapid releases.

The results also have demonstrated that most of the issues that can be found in the literature are present in the industry too. The most critical problem found at the industry is having flaky tests. Also, time-consuming testing, GUI automated testing and big data testing are high severity issues.

The findings of this survey might contribute to new research lines on software quality. It is therefore that in future works deeper research will be carrying on, focusing on the most critical testing problems in rapid releases.

Acknowledgment

This paper was developed in the context of a PhD tesis on computer science and it has been supported by the projects "Metodologías y herramientas emergentes para contribuir con la calidad del software" (PI 17F018 SCyT UNNE) and "Análisis e

Implementación de tecnologías emergentes en sistemas computacionales de aplicación regional" (PI 17F017 SCyT UNNE).

We also thank to the 255 respondents of the survey for their contribution to this study.

References

1. Humble, J., Farley, D.: Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Pearson Professional, Upper Saddle River, NJ (2010).
2. Duvall, P., Matyas, S., Glover, A.: Continuous integration: Improving software quality and reducing risk. Pearson Professional, Upper Saddle River, NJ (2007).
3. Fowler, M.: Continuous Integration, https://martinfowler.com/articles/continuousIntegration.html.
4. Fowler, M.: Continuous Delivery, https://martinfowler.com/bliki/ContinuousDelivery.html.
5. Chen, L.: Continuous Delivery at Scale: Challenges and Opportunities. In: 2018 IEEE/ACM 4th International Workshop on Rapid Continuous Software Engineering (RCoSE). pp. 42–42 (2018).
6. Chen, L.: Continuous Delivery: Huge Benefits, but Challenges Too. IEEE Software. 32, 50–54 (2015). https://doi.org/10.1109/MS.2015.27.
7. Chen, L.: Continuous Delivery: Overcoming adoption challenges. Journal of Systems and Software. 128, 72–86 (2017). https://doi.org/10.1016/j.jss.2017.02.013.
8. Neely, S., Stolt, S.: Continuous Delivery? Easy! Just change everything (well, maybe it is not that easy). In: 2013 Agile Conference. pp. 121–128 (2013). https://doi.org/10.1109/AGILE.2013.17.
9. Fitzgerald, B., Stol, K.-J.: Continuous software engineering and beyond: Trends and challenges. In: Proceedings of the 1st International Workshop on Rapid Continuous Software Engineering. pp. 1–9. ACM, New York, NY, USA (2014). https://doi.org/10.1145/2593812.2593813.
10. Shahin, M., Babar, M.A., Zhu, L.: Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices. IEEE Access. 5, 3909–3943 (2017). https://doi.org/10.1109/ACCESS.2017.2685629.
11. Laukkanen, E., Itkonen, J., Lassenius, C.: Problems, causes and solutions when adopting continuous delivery — A systematic literature review. Information and Software Technology. 82, 55–79 (2017). https://doi.org/10.1016/j.infsof.2016.10.001.
12. Shankland, S.: Rapid-release Firefox meets corporate backlash, https://www.cnet.com/news/rapid-release-firefox-meets-corporate-backlash/.
13. Kaply, M.: Why Do Companies Stay on Old Technology?, https://mike.kaply.com/2011/06/23/why-do-companies-stay-on-old-technology/.
14. Khomh, F., Dhaliwal, T., Zou, Y., Adams, B.: Do faster releases improve software quality? An empirical case study of Mozilla Firefox. In: 2012 9th IEEE Working Conference on Mining Software Repositories (MSR). pp. 179–188 (2012). https://doi.org/10.1109/MSR.2012.6224279.
15. Porter, A., Yilmaz, C., Memon, A.M., Krishna, A.S., Schmidt, D.C., Gokhale, A.: Techniques and processes for improving the quality and performance of open-source software. Software Process: Improvement and Practice. 11, 163–176 (2006). https://doi.org/10.1002/spip.260.
16. Baysal, O., Davis, I., Godfrey, M.W.: A Tale of Two Browsers. In: Proceedings of the 8th Working Conference on Mining Software Repositories. pp. 238–241. ACM, New York, NY, USA (2011). https://doi.org/10.1145/1985441.1985481.

17. Mäntylä, M.V., Khomh, F., Adams, B., Engström, E., Petersen, K.: On Rapid Releases and Software Testing. In: 2013 IEEE International Conference on Software Maintenance. pp. 20–29 (2013). https://doi.org/10.1109/ICSM.2013.13.
18. Mascheroni, M.A., Irrazábal, E.: Problemas que afectan a la calidad de software en entrega continua y pruebas continuas. Presented at the XXIV Congreso Argentino de Ciencias de la Computación (La Plata, 2018). (2018).
19. Mascheroni, M.A., Irrazábal, E.: Continuous Testing and Solutions for Testing Problems in Continuous Delivery: A Systematic Literature Review. Computación y Sistemas. 22, (2018). https://doi.org/10.13053/cys-22-3-2794.
20. Pfleeger, S.L., Kitchenham, B.A.: Principles of Survey Research: Part 1: Turning Lemons into Lemonade. SIGSOFT Softw. Eng. Notes. 26, 16–18 (2001). https://doi.org/10.1145/505532.505535.
21. Kitchenham, B.A., Pfleeger, S.L.: Principles of Survey Research Part 2: Designing a Survey. SIGSOFT Softw. Eng. Notes. 27, 18–20 (2002). https://doi.org/10.1145/566493.566495.
22. Kitchenham, B.A., Pfleeger, S.L.: Principles of Survey Research: Part 3: Constructing a Survey Instrument. SIGSOFT Softw. Eng. Notes. 27, 20–24 (2002). https://doi.org/10.1145/511152.511155.
23. Kitchenham, B., Pfleeger, S.L.: Principles of Survey Research Part 4: Questionnaire Evaluation. SIGSOFT Softw. Eng. Notes. 27, 20–23 (2002). https://doi.org/10.1145/638574.638580.
24. Kitchenham, B., Pfleeger, S.L.: Principles of Survey Research: Part 5: Populations and Samples. SIGSOFT Softw. Eng. Notes. 27, 17–20 (2002). https://doi.org/10.1145/571681.571686.
25. Kitchenham, B., Pfleeger, S.L.: Principles of Survey Research Part 6: Data Analysis. SIGSOFT Softw. Eng. Notes. 28, 24–27 (2003). https://doi.org/10.1145/638750.638758.
26. Kitchenham, B.A., Pfleeger, S.L.: Personal Opinion Surveys. In: Shull, F., Singer, J., and Sjøberg, D.I.K. (eds.) Guide to Advanced Empirical Software Engineering. pp. 63–92. Springer London, London (2008). https://doi.org/10.1007/978-1-84800-044-5_3.
27. Koschke, R.: Software Visualization in Software Maintenance, Reverse Engineering, and Re-engineering: A Research Survey. Journal of Software Maintenance. 15, 87–109 (2003). https://doi.org/10.1002/smr.270.
28. Chow, T., Cao, D.-B.: A survey study of critical success factors in agile software projects. Journal of Systems and Software. 81, 961–971 (2008). https://doi.org/10.1016/j.jss.2007.08.020.
29. Lethbridge, T.C.: A Survey of the Relevance of Computer Science and Software Engineering Education. In: Proceedings of the 11th Conference on Software Engineering Education and Training. pp. 0056–. IEEE Computer Society, Washington, DC, USA (1998).
30. R. VIJAYASARATHY, L., TURK, D.: Agile software development: A survey of early adopters. Journal of Information Technology Management. 19, (2008).
31. Garousi, V., Zhi, J.: A survey of software testing practices in Canada. Journal of Systems and Software. 86, 1354–1376 (2013). https://doi.org/10.1016/j.jss.2012.12.051.
32. Mao, K., Capra, L., Harman, M., Jia, Y.: A survey of the use of crowdsourcing in software engineering. Journal of Systems and Software. 126, 57–84 (2017). https://doi.org/10.1016/j.jss.2016.09.015.
33. Yang, L.-R., Huang, C.-F., Wu, K.-S.: The association among project manager's leadership style, teamwork and project success. International Journal of Project Management. 29, 258–267 (2011). https://doi.org/10.1016/j.ijproman.2010.03.006.
34. T. O'Connor, J., Yang, L.-R.: Project Performance versus Use of Technologies at Project and Phase Levels. Journal of Construction Engineering and Management-asce - J CONSTR ENG MANAGE-ASCE. 130, (2004). https://doi.org/10.1061/(ASCE)0733-9364(2004)130:3(322).